

CS411 Final Project Report

Spring 2019

Jiaying Wu (jwu86)

Siqi Zhang (siqi5)

Baohe Zhang (baohez2)

Yuying Chen (yuyingc3)

1. Brief description

We build a movie playground for movie fans. Users can view basic information about movies, for example, posters, overview, ratings, etc. Also, users can search for movies by keywords in the title, or find lists of movies by selecting movie genres. The great feature about our platform is that users can rate and write reviews for movies they have watched. Our system can then provide personalized recommendations based on users' "likes", i.e. rating distribution and similar interests shared among user groups. The review part of the movies is an awesome place for users to share their feedbacks of the movie, a micro-community for open discussions.

Our recommendation mechanism is based on the user's likes and similar interests shared among user groups. For example, you enjoy animation movies a lot, and you rate movies tagged of "animation" rather higher than other genres, then our system will recommend the movies that are popular among users who also rate "animation" movies high a lot. The idea of the user group is that, if user A and B demonstrate similar interests in the past, e.g. they rated the same movie at roughly similar ratings, then there's a large chance that they will enjoy similar movies later, thus our system will propose the movie A like to B.

On the user portal page, users can see the ratings and reviews they gave. From the homepage, the user can go to a movie detail page by clicking on the poster of that movie or search for a movie by some keywords. The user can Sign-in or Register to set up an account. Our rating and review functions are only open to signed-in users.

2. Usefulness

Enjoyed a movie? Then give it a five-star score and write some review. Our movie recommendation system will then know more about your preferences over time and try to feed you with a list of movies that you will enjoy watching next time.

When you have some movie in mind and don't know the exact title, you can just type in some keywords and see if there's a match. When you don't have any idea and just want to watch a movie, you can look into our personalized recommendation based on your watching history and the preference in your user groups. A user group is home to those that share similar ratings on a rather large number of movies. After watching a movie, leave a rating, preferably a comment, because this will record your footprint and help us better recommend movies for all users.

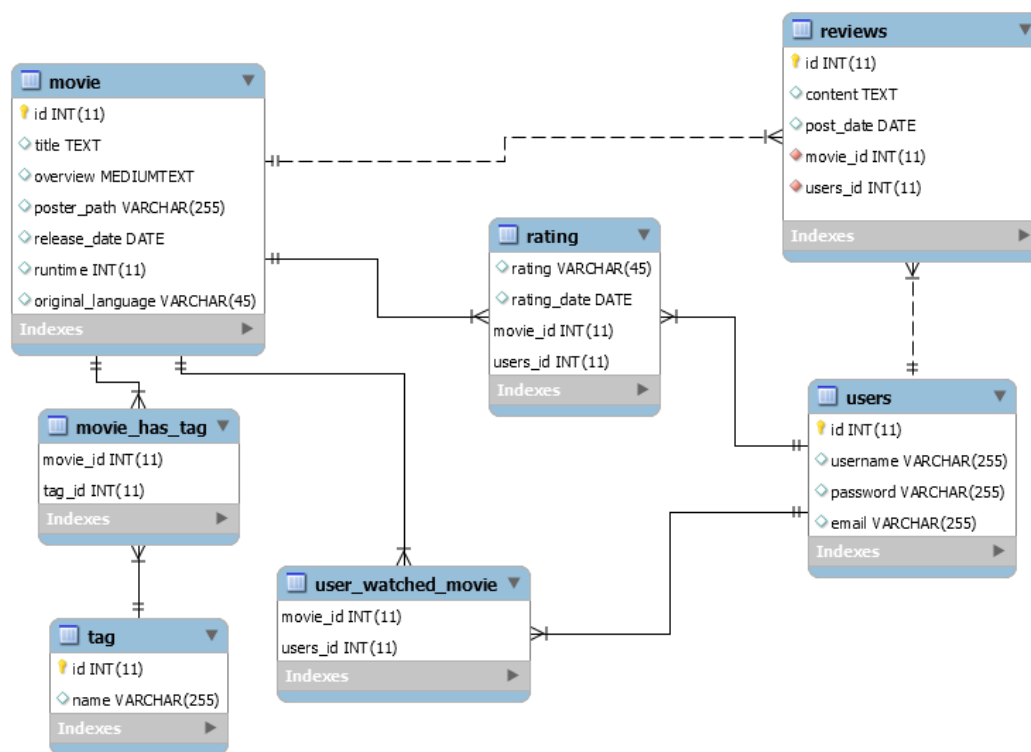
The main page of our system will demonstrate movies and their posters based on popularity and ratings. You can never go wrong on those hot hits.

3. Discuss the data in your database

We will use the Movie Database (TMDB). The dataset contains the Primary info, Alternative titles, Cast, Crew, Images (posters, backdrops), Plot keywords, Release information, Trailers, Translations, Similar movies, Reviews, Belongs to lists, Changes. The data is fetched from the APIs and stored into MySQL database using JSON.

User information is created when they sign up, rate movies, and write reviews.

4. ER Diagram and Schema



5. Data Collection and Crawling

The movie information and reviews are collected from TMDB. These information are collected and saved by some PYTHON scripts using TMDB APIs and turn into MySQL inserting script. The user information and rating are created by ourselves.

6. Clearly list the functionality of your application (feature specs)

Basic CRUD.

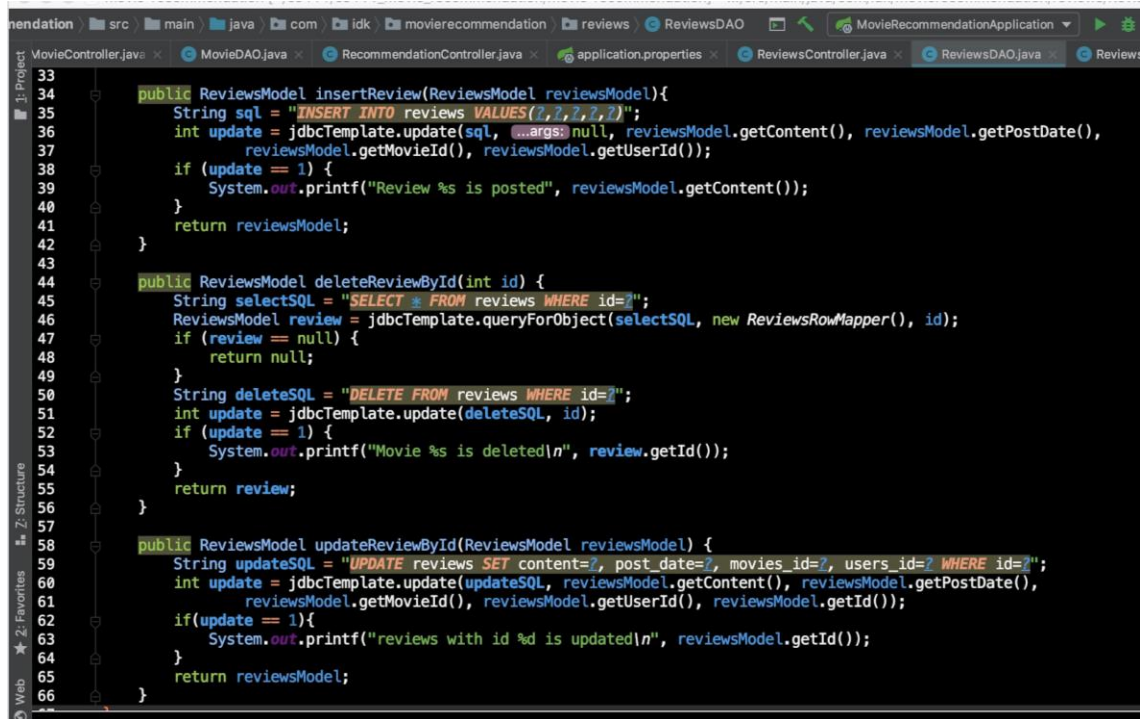
- Users look for movies on the home page by searching for keywords.
- Users can choose to write reviews for movies, creating an open discussion forum for movies.
- One thing that users will refer to when they want to decide whether to see a movie is ratings. Our system provides rating functionality, users rate movies on a basis of 1 - 5 stars on the movie detail page.
- User portal. Users sign up by emails and can manage, modify and delete, their comments on their personal page.

7. One basic function

The user platform of our system provides a place for users to see all the reviews they have written, modify or delete if they want. They write the reviews on the movie detail page under the movie information part and manage the reviews on their personal page. In the review table, there are 5 attributes, the id being the primary key to identify review entries, and the others being movies_id, users_id, post_date, and content. Example rows in the review table are shown below.

id	content	post_date	movies_id	users_id
119	I'm at a total loss. I do reviews for friends and fa...	2019-04-10	0	0
120	So, I had hoped this was going to be a good mo...	2019-04-10	0	1
121	Weird seing a movie like this in the cinema. Isn't...	2019-04-16	0	2
122	I expected more than this. Everything is just so...	2019-04-18	0	5
123	Feeling more like Sam Raimi's Spiderman than...	2019-04-16	0	4
124	Terrible movie bad acting by brie larson and a lo...	2019-03-09	1	0
125	Characters - Were boring and forgettable. Cine...	2019-04-01	1	1

When users insert reviews, the system first gets the movie which users want to write review for and their user id, then insert the review entry to the review table with all the five required attributes mentioned above, i.e., post_data, movies_id, user_id, content and id(for each unique review).



```
33
34 public ReviewsModel insertReview(ReviewsModel reviewsModel){
35     String sql = "INSERT INTO reviews VALUES(?, ?, ?, ?, ?)";
36     int update = jdbcTemplate.update(sql, reviewsModel.getMovieId(), reviewsModel.getPostDate(),
37     reviewsModel.getContent(), reviewsModel.getUserId());
38     if (update == 1) {
39         System.out.printf("Review %s is posted", reviewsModel.getContent());
40     }
41     return reviewsModel;
42 }
43
44 public ReviewsModel deleteReviewById(int id) {
45     String selectSQL = "SELECT * FROM reviews WHERE id=?";
46     ReviewsModel review = jdbcTemplate.queryForObject(selectSQL, new ReviewsRowMapper(), id);
47     if (review == null) {
48         return null;
49     }
50     String deleteSQL = "DELETE FROM reviews WHERE id=?";
51     int update = jdbcTemplate.update(deleteSQL, id);
52     if (update == 1) {
53         System.out.printf("Movie %s is deleted\n", review.getId());
54     }
55     return review;
56 }
57
58 public ReviewsModel updateReviewById(ReviewsModel reviewsModel) {
59     String updateSQL = "UPDATE reviews SET content=?, post_date=?, movies_id=?, users_id=? WHERE id=?";
60     int update = jdbcTemplate.update(updateSQL, reviewsModel.getContent(), reviewsModel.getPostDate(),
61     reviewsModel.getMovieId(), reviewsModel.getUserId(), reviewsModel.getId());
62     if(update == 1){
63         System.out.printf("reviews with id %d is updated\n", reviewsModel.getId());
64     }
65     return reviewsModel;
66 }
```

When users are at their user portal, they can see all the reviews they have written, which is implemented by the following code snippet. Then they can do the modify and deletion of the reviews shown.

```
public List<ReviewsModel> getReviewById(int id){
    String sql = "SELECT * FROM reviews WHERE id=?";
    List<ReviewsModel> reviewList = jdbcTemplate.query(sql, new ReviewsRowMapper(), id);
    return reviewList;
}
```

8. SQL code snippet

CRUD of movies.

```

public MovieModel insertMovie(MovieModel movie) {
    String sql = "INSERT INTO movie VALUES(?, ?, ?, ?, ?, ?, ?)";
    int update = jdbcTemplate.update(sql, movie.getId(), movie.getTitle(), movie.getOverview(), movie.getPosterPath(),
        movie.getReleaseDate(), movie.getRuntime(), movie.getOriginalLanguage());
    if (update == 1) {
        System.out.printf("Movie %s is inserted\n", movie.getTitle());
    }
    return movie;
}

public MovieModel deleteMovieById(int id) {
    String selectSQL = "SELECT * FROM movie WHERE id=?";
    MovieModel movie = jdbcTemplate.queryForObject(selectSQL, new MovieRowMapper(), id);
    if (movie == null) {
        return null;
    }
    String deleteSQL = "DELETE FROM movie WHERE id=?";
    int update = jdbcTemplate.update(deleteSQL, id);
    if (update == 1) {
        System.out.printf("Movie %s is deleted\n", movie.getTitle());
    }
    return movie;
}

```

```

public MovieModel getMovieById(int id) {
    String sql = "SELECT * FROM movie WHERE id=?";
    MovieModel movie = jdbcTemplate.queryForObject(sql, new MovieRowMapper(), id);
    return movie;
}

```

```

public MovieModel updateMovie(MovieModel movie){
    String updateSQL = "UPDATE movie SET title=?, overview=?, poster_path=?, release_date=?, runtime=?, original_language=? WHERE id=?";
    int update = jdbcTemplate.update(updateSQL, movie.getTitle(), movie.getOverview(), movie.getPosterPath(), movie.getReleaseDate(),
        movie.getRuntime(), movie.getOriginalLanguage(), movie.getId());
    if(update == 1){
        System.out.printf("Movie %s is updated\n", movie.getTitle());
    }
    return movie;
}

```

Helper functions for movie operations.

```

public Integer getMovieNumber() {
    String sql = "SELECT COUNT(*) AS number FROM movie";
    Integer number = jdbcTemplate.queryForObject(sql, new Object[] {}, Integer.class);
    return number;
}

public List<MovieModel> getMoviesByTag(int tagId) {
    String sql = "SELECT movie.id, movie.title, movie.overview, movie.poster_path, movie.release_date, movie.runtime, " +
        "movie.original_language " +
        "FROM movie, movie_has_tag " +
        "WHERE movie.id = movie_has_tag.movie_id AND movie_has_tag.tag_id = ?";
    List<MovieModel> moviesList = jdbcTemplate.query(sql, new MovieRowMapper(), tagId);
    return moviesList;
}

public MovieModel getMovieByName(String moviename){
    String sql = "SELECT * FROM movie WHERE title=?";
    MovieModel ret = jdbcTemplate.queryForObject(sql, new MovieRowMapper(), moviename);
    return ret;
}

public List<MovieModel> getMatchMovies(String name){
    String sql = "SELECT * FROM movie WHERE title LIKE ?";
    return jdbcTemplate.query(sql, new MovieRowMapper(), ...args: "%" + name + "%");
}

```

Snippets for the review function are shown and discussed in part 7.

```
@Repository
public class UserTagCountDAO {

    @Autowired
    private JdbcTemplate jdbcTemplate;

    public List<UserTagCountModel> getUserTagCountList(String username) {
        String sql = "SELECT movie_has_tag.tag_id, COUNT(movie_has_tag.tag_id) " +
            "FROM users, user_watched_movie, movie_has_tag " +
            "WHERE users.username = ? AND users.id = user_watched_movie.user_id " +
            "AND movie_has_tag.movie_id = user_watched_movie.movie_id " +
            "GROUP BY movie_has_tag.tag_id;";

        List<UserTagCountModel> userTagCountList = jdbcTemplate.query(sql, new UserTagCountRowMapper(), username);
        return userTagCountList;
    }
}
```

This query is used to find the count of each tag for a specific user based on the user's watched history.

9. List and briefly explain the dataflow, i.e. the steps that occur between a user entering the data on the screen and the output that occurs (you can insert a set of screenshots)

User Register and Login:

Input the information of a new user => Call a post to propagate user information to backend
=> Insert a new row in user table in database => log in => retrieve the user information =>
send a get to the backend => check correctness of password at front end

Welcome

Register

Welcome

Login

SignUp

Invalid Username or Password

Movie information:

Click movie at main page => jump to the url point to the specific movie => retrieve movie id from url => send 2 get requests to backend => get the details of movie and review list for the movie in database => return json of movie details to front end and render



Captain Marvel

Overview

The story follows Carol Danvers as she becomes one of the universe's most powerful heroes when Earth is caught in the middle of a galactic war between two alien races. Set in the 1990s, Captain Marvel is an all-new adventure from a previously unseen period in the history of the Marvel Cinematic Universe.

Genres:

Action Adventure Science Fiction

Score: ★ ★ ★ ★ ★

Date: 2019-03-06

Language: en

Runtime: 124 minutes

Review posting and show:

Check if the user log in =>

If not, the review component will not be activate.

Please Login to Review

If yes => when put a new review, send a post to backend and insert a new review into database => update the review lists show in single page.

Review

Submit

Review update and delete:

MovieStudio Category Trend Recommend <input type="text"/> <input type="button" value="Search"/>				Review	Login
My Movie Reviews					
review	date	Update	Delete		
So, I had hoped this was going to be a good movie and parts of it we're but other parts weren't. First off, while the movie had very strong family elements I wouldn't take my children at all. There villain's almost added a element of horror to the film. One specific scene 7 demons are unleashed upon a board room of people. One of the demons lifts a guy up bites his head off then throws him out the window. This scene lasts for a while as you hear and see people begging for their lives as the demons kill them one by one. This would have definitely disturbed elementary school me. Also the main character Billy Batson has had a character change from comic to film. What I mean is, I really didn't like how they made him much more selfish during most of this movie. I personally like the fact that he is a good kid that tries to always do the right thing. Now for this story I see how they used this character flaw to help him overcome it in the end making him a true hero after all but I would have come up with a different character flaw to overcome. Last and my biggest hurdle to try to overcome, I did not and do not like the actor they picked to play Shazam/Captain Marvel. First off, the character had to use a muscle suit in order to look buff. Second the over all look and behavior of the actor killed it for me, I couldn't take him seriously. He reminded me to much of Jimmy Fallon throughout the whole film who even though is a funny guy does not fit the bill to play this hero. I just couldn't get behind the character. I wanted to like this one cause personally I really like the character from the comics but this movie fell short for me.	2019-04-10	<input type="button" value="Update"/>	<input type="button" value="Delete"/>		
Characters - Were boring and forgettable. Cinematography - Had one good shot and the rest of the film was basic Nick Fury - Made as a joke, unintelligent and pointless. Captain Marvel - No tension felt for her as she is overpowered and emotionless. And learns nothing throughout this film and is still the same arrogant character. Writing - The writing was dreadful and missed emotional opportunities to really make us feel for the characters, it felt like a child written it. Brie Larson - The wrong person casted as this character. She's a good actor, don't get me wrong but she	2019-04-01	<input type="button" value="Update"/>	<input type="button" value="Delete"/>		

Click on Review on navigation bar => retrieve user information => send a get request to backend => get a review list written by the specific user => render at front end.

Click on delete => send a delete request to the backend => delete the review from database

Click on update => jump to the page to write a new review => send a put request to backend => update the review in database => jump to previous review page

My Review

Rating:

Check if the user log in =>

If not, the rating component is gray, a new rate can't be post.

If yes => retrieve user information and send a get request to get the user's rating for this movie => If the return rate is not empty, show user's rate. If the return is null, enable the

component to post a new rate => when put a new rate, send a post to backend and insert a new rate into database => update the average rate of this movie

10. two advanced functions.

There are two advanced functions in our movie recommendation website. The first one is to recommend the movies to users based on their watch history and tag of movie. The second advanced function is to show the customized review word cloud to user based on user similarity.

(1) personalized movie recommendation

Our personalized movie recommendation system can generate a list of recommendation movies based on the user's "watched" history. The "watched" history is collected by the user's rating and reviewing history. There are three steps to generate recommendation movies. The first step is to calculate the user preference vector according to the user's watching history. The user preference vector is a vector of size N , where N stands for the total number of tags we generalizing for all movies. The value at a specific index of the vector represents how much the user like this tag. The second step is to retrieve a movie * tag matrix, where each cell represents a containing relationship between a movie and a tag. The shape of the matrix is $M * N$, where M represents the total number of movies and N represents the total number of tags. For example, $matrix[i, j] = 1$ means the i -th movie contains j -th tag. The third step is to multiply each row in the matrix with the user preference vector and obtain top K movies as the recommendation movies for that user.

This function can be considered as advanced because the movie recommendation is an on-the-fly calculation process, and this calculation involves matrix multiplication. The recommendation movies for a user is dynamic, and those can be changed when the user's preference is changed.

(2) Customized Review Word Cloud

Our Customized Review Word Cloud will show user key words from reviews which are written by reviewers who have similar tastes as the user. There are two main parts of this advanced function.

One is user similarity measurement. We compute user similarity score by constructing user rating matrix. In our database, we have a table called ratings which contains `user_id`, `movie_id` and `rating` as attributes. Suppose the number of unique users in ratings table is m and number of unique movies in ratings is n , we will construct an $m*n$ matrix. Each entry is the rating of user i on movie j . We

filling the information by querying the ratings table. After constructing the matrix, we treat each row as a user vector and compute the cosine similarity between user vectors. Given a target user x who is currently logged in our system, we will calculate the similarity score between the target user and other users in our rankings table and rank them from most similar to least similar. For each user that was compared with target user we will assign a significance weight to each of them. The most similar user (who get rank one in vector similarity comparison) will get a weight 2.0 and the rank r user will get a weight $2.0 \cdot (0.8^r)$.

The next step is to prepare word cloud input. Suppose user x want to see the word cloud of movie y , we will query all reviews on movie y from reviews table. For every single review, we know the reviewer_id and content of the review. In order to get the counts of every keyword in the reviews, we need to preprocess the reviews first. We simply use split by space method to tokenize the reviews and then remove the stop words from reviews. We tried TF-IDF to help with stopword removal, but it removes some good word as well, so we finally choose to maintain a fixed stop word list to help the removal step. We now get a list of keywords from reviews about movie y and we know which word comes from which review and which review is written by which reviewer. For each unique word u in the review keywords list, we know u is mentioned by a set of reviewers s on movie y . We set the customized frequency of u to be the summation of rank r reviewer's weight, for reviewers who is in s and send the frequency map to frontend to show it. By such calculation, we are able to give reviews of reviewers who have similar taste as user x more weight and give reviews of reviewers who have different taste as user x less weight, so our cloud map is able to highlight the reviews of other users having similar taste as the current target user, and makes their keywords in the review content stand out, i.e., more eye catching in the cloud map.

This function is considered as advanced in two aspects. In the technique aspect, this function cannot be accomplished by simply querying the database; it has to go through some complex post processing in the backend. When we build this function, we did not call any API in the backend, and this function involves some simple natural language processing and text mining techniques. In the user aspect, it uses user rating as a feature and uses rating based similarity measurement to provide a better user experience. From these two points, Customized Review Word Cloud is an advanced function.

A sample cloud map of Avengers is shown below. The first one is the common word map that everyone can see, and the second one is when a target user, say Thanos, sees when he has logged in. We can see that Thanos gave the movie, Avengers: Infinity War, a four star rating, and so his cloud map gives more weights to the keywords from the reviews written by users who gives an around 4 stars to this movie, and gives similar ratings to other movies as Thanos.

two functions are necessary in our design, though both of them serves recommendation purpose. If we only have the first advanced function, it is possible that the movie genre is what user likes, but the movie content or cast doesn't fit user's taste. By adding second advanced function, user is able to get a recommendation from a different angle and have a better sense whether he or she will watch it. If we only have the second function, in order to find the recommended movie by scanning the important word in review, the user have to go through the whole movie list. Therefore, in order to have a better user experience, we use the first advanced function to serve as a broad range filter, and use the second advanced function to provide more precise insight recommendation of the movie.

11. Describe one technical challenge that the team encountered?!This should be sufficiently detailed such that another future team could use this as helpful advice if they were to start a similar project or were to maintain your project. Say you created a very robust crawler - share your knowledge. You learnt how to visualize a graph and make an interactive interface for it - teach all! Know how to minimize time building a mobile app - describe

We figure a problem in initialize a page with ReactJS. The page for a single movie has to do lots of initialize in function componentDidMount, including getting the movie details, retrieving user information, getting all the ratings and reviews for the movie, and get the rating put by the specific user.

The problem is, there could be no logging in, so the user information can be empty. Even when the user information can be retrieved successfully, the rating for the movie can be empty. So when initializing, some parts can be suspend, and the next won't be execute. The initializing result is different than we expected.

Finally, we use some if-else in the initialize part to avoid the suspend. For example, we only send get specific rating result when the retrieved user information is not null. And we get the tags information only after getting correct movie information. This part of code is as below.

```

if (Authentication.isUserLoggedIn()) {
  UserReviewService.retrieveUserInfo(username).then(res => {
    const userInfo = res.data;
    vote.user_id = userInfo.id;
    MovieService.getRatingByUserAndMovie(vote.user_id, vote.movie_id).then(
      res => {
        vote.rating = res.data.rating;
        vote.rating_date = res.data.rating;
        this.setState({ vote });
      }
    );
  });
}

```

12. State if everything went according to the initial development plan and proposed specifications, if not - why?!

We went everything almost same as we planned. We accomplish the basic functionality as described in our plan and we finish most part of advanced functionality. The planned advanced functionality have four part: advanced, search, review customized word cloud visualization, personalized recommendations, and fraud detection. The two things we haven't accomplish are Fraud Detection and Advanced Search. We plan to make advanced search in our backend, and user can get the movie by just typing any keyword such as genre and cast in search box. However, due to time limit, we are only able to allow user to type movie title and show up most similar movie titles according to user's typing input. This process is done in frontend rather than backend. Regarding to the fraud detection, we considering the fact that if we just simply remove the users whose rating is always too low or always too high will remove some actual user and our time limitation doesn't allow us to come up with a better idea, so we didn't accomplish that.

13. Describe the final division of labor and how did you manage team work.

We divide the word to be main parts, and for each part we precisely divide the work to backend and frontend. The precise labor division is listed below:

Content	FrontEnd/BackEnd	Group Member Name
Movie Main Page	FrontEnd	Baohe Zhang
	BackEnd	Baohe Zhang

Single Movie Page	FrontEnd	Yuying Chen, Siqi Zhang
	BackEnd	Siqi Zhang
User Review Page	FrontEnd	Jiaying Wu
	BackEnd	Jiaying Wu
Movie Rating	FrontEnd	Siqi Zhang, Yuying Chen
	BackEnd	Yuying Chen
Single Movie Review	FrontEnd	Yuying Chen
	BackEnd	Yuying Chen
User Login, Logout, Registration	FrontEnd	Jiaying Wu
	BackEnd	Jiaying Wu
Advance function1: Movie Recommendation	FrontEnd	Baohe Zhang
	BackEnd	Baohe Zhang
Advance function2: Review Cloud Map	FrontEnd	Jiaying Wu
	BackEnd	Jiaying Wu
Code Merge		Yuying Chen, Baohe Zhang
Search Auto-Complete	FrontEnd	Siqi Zhang

Our team have weekly meeting. In every meeting, we merge the code and discuss the work division in next week.